Learning user preferences for participatory sensing tasks

Student Name: Carlos Josue Moz Preza

SID: 3301150005

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Information and Communication Systems

OCTOBER 2016

THESSALONIKI – GREECE

INTERNATIONAL HELLENIC UNIVERSITY

Learning user preferences for participatory sensing tasks

Student Name: Carlos Josue Moz Preza

SID: 3301150005

Supervisor:	Dr. Merkouris Karaliopoulos
Supervising Committee Members:	Assoc. Prof. Name Surname
	Assist. Prof. Name Surname

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Information and Communication Systems

Acknowledgments

I would like to express my deepest gratitude to my supervisor Dr. Merkouris Karaliopoulos. His valuable advice and our frequent communication along the research project helped me clarify and understand better the problems faced during the dissertation process.

I would also like to thank to Dr. Marios Gatzianas, who has assisted with forwarding the questionnaire to the appropriate mailing lists of the IHU, i.e., students and alumni.

Last but not least, I would like to thank my parents and brothers, who have been supporting me along my entire life. This thesis is dedicated to them, for their infinite love.

Abstract

Nowadays, we experience the rapid proliferation of smartphones and wearable technologies, with several embedded sensors that are capable to sense humidity, temperature, light, and proximity among others. This leads to a new pervasive service paradigm called participatory sensing (or, often interchangeably, mobile crowdsensing). Participatory sensing leverages the power of the crowds in that, end users can collect sensing data through their smart mobile devices, and contribute them to a central platform that processes them to build services out of them. Therefore, new applications and services can be generated out of the collective effort of many users, some of which might be totally infeasible or far more resource demanding otherwise. However, to ensure the sustained participation of end users in these services, it is important to provide them with proper incentives for their contributions. These can be either monetary or non-monetary; in any case, it is mandatory to tailor these incentives to the users' individual preferences.

This dissertation's objective is to explore new ways of inferring and subsequently modeling the individual user preferences in the context of participatory sensing applications. To this end, it draws on historical data from the interaction of the end users with the application and applies machine learning techniques to efficiently profile users. The ultimate aim is to take advantage of these user profiles in the process of targeting incentives to them.

The project research work is divided in two phases. The first phase is concerned with the data collection process. In the absence of data from real crowdsensing applications, the relevant data for the proof-of-concept experimentation is collected through an online questionnaire. The questionnaire is addressed to students at IHU and also uploaded to special-purpose websites that crowdsource responses to such research efforts. A total of 132 user responses was collected during this first phase of the project.

Then, in the second phase of the research work we apply machine learning models (specifically: logistic regression models), to infer the individual end user preferences and analyze the similarity/ diversity characterizing them. More specifically, the choice problem users face when presented with multiple offers for participatory sensing tasks is captured as an instance of multi-attribute decision making problems with multiple alternatives and modeled through probabilistic multi-class logistic regression models. Clustering and

community detection techniques are used to identify similarity and diversity trends in these preferences, with the aim to specify “classes” of users with distinct preference features.

The outcome of this work, the user models and the related accuracy scores together with the classes that segregate the individual user preferences, form a major part of a paper that will be submitted to IEEE Transaction on Mobile Computing. The paper builds on the models derived in this Dissertation to analytically optimize the offered incentives to participatory sensing users.

Carlos Josue Moz Preza

December 19th, 2016

Contents

List of Figures

List of Tables

1 Background/motivation

The dissertation essentially revolves around the participatory sensing paradigm, often interchangeably called mobile crowdsensing.

1.1 Participatory sensing

The proliferation of smartphones and wearable technology products, with various embedded sensors such as accelerometers, GPS, video and image sensors, acoustic sensors, gyroscope and proximity sensors have motivated a new service paradigm. Equipped with what could be seen as mobile sensor platforms, end users have the capability to sense, process and share local knowledge on the move [1]. Data that can be shared include measurements (e.g., of humidity, noise, temperature), pictures, messages, and even video files. The variety in the types of collected and shared data is further enlarged through new sensor types that can be interfaced with the smartphones via Bluetooth or wired media.

Several participatory sensing applications have emerged over the last decade. One type of such applications is concerned with environmental monitoring. The collected data relates to measures of environmental quantities (e.g., samples of of noise, air pollution, temperature, or humidity from different locations across an area). Out of these samples, maps can be generated depicting how the particular measure varies spatially across an area that can be as large as city. Such applications have a strong community dimension: users contribute data that eventually generate a service that is valuable for themselves. An example of such application is NoiseTube, a project that aims at generating noise maps in areas that are particularly burdened with noise. End users can download an application that activates their microphone whenever they lie within areas of interest. The microphone collects noise samples and transmits them to a backend serves that builds the map. The quality of the samples varies depending on the smartphone and microphone quality and the sampling conditions (e.g., whether the smartphone is within a bag, the user's pocket or held in the air).

Another broad family of applications relates to health and fitness. These apps help monitor and report data about personal activities, and then share them and compare with others

(*e.g.*, body/mass indices, fat, food habits, sleep). Through this sharing, it is possible to cross-relate data and better understand personal health, in order to provide recommendations/advice about factors affecting health. An example of such an application is the mobile app by Dacadoo. The app collects a combination of data, from physiometric indices to usage context ones down to replies of users to questionnaires for an interval of time (typically one week). It then computes a fitness score for the app users and lets friends or colleagues compete with each other.

Transportation-related applications such as Waze and a variety of social parking apps sense information about traffic and parking space status, respectively. These types of applications provide maps of traffic across the city in real time, as well as parkings spaces available. In some cases, they even let drivers handover parking sports between drivers. Hence, they inform the choices of drivers about smarter routes and faster parking search in the city, which end up reducing fuel consumption, time waste, and CO2 emissions.

(Semi-) real time information sharing is the main features of urban sensing applications (sometimes, also called social journalism type applications). They provide a means to quickly disseminate information related to things happening in the city/neighborhood (e.g., Improve-my-city) such as events/problems/opportunities.

Finally, there is a plethora of commercial applications, beyond community-driven, where end users contribute information and data that are not necessarily of personal interest, in return for some monetary reward. The monitoring of automatic vendor machines or garbage bins and the photo-shoots of food and dining places for marketing purposes are examples of applications, where crowds of mobile users, typically with better-than-average skills, are chosen to carry out a task that otherwise would be carried out by a professional at higher cost.

One way to be thinking about these applications is shown in Figure 1.1. Their typical functionality can be organized across three different layers. The bottom layer contains all those operations that facilitate the collection of data *from* the crowd of end users, including the matching of tasks with users and the provision of proper incentives to motivate their contributions. Part of this layer is also the implementation of privacy measures that let end users contribute to MCS tasks without sacrificing what they perceive as private information.

Then there is the top layer that collects the contributions of the crowd and turns them into meaningful services. Often termed the collective intelligence layer, it involves tasks that may vary broadly and are application-specific (*e.g.*, event mapping). The resulting services also vary significantly, from informative maps (*e.g.*, pollution or noise maps) to personal assistants (*e.g.*, applications that help choosing congestion-free routes or find parking space faster while driving across the city).

Finally, almost always there is a third layer that is less visible but critical for the application. This layer deals with the collection of data *about* the application users: their mobility patterns, preferences, and past responses to task offers/assignments. Outcomes of this layer, which draws heavily on data analytics, are dynamically updated user profiles that inform most of the application's operations. One example, which is the main focus of this dissertation, is to analyze these profiles to identify tasks that are more attractive for individual users and tailor the incentives offered to them.

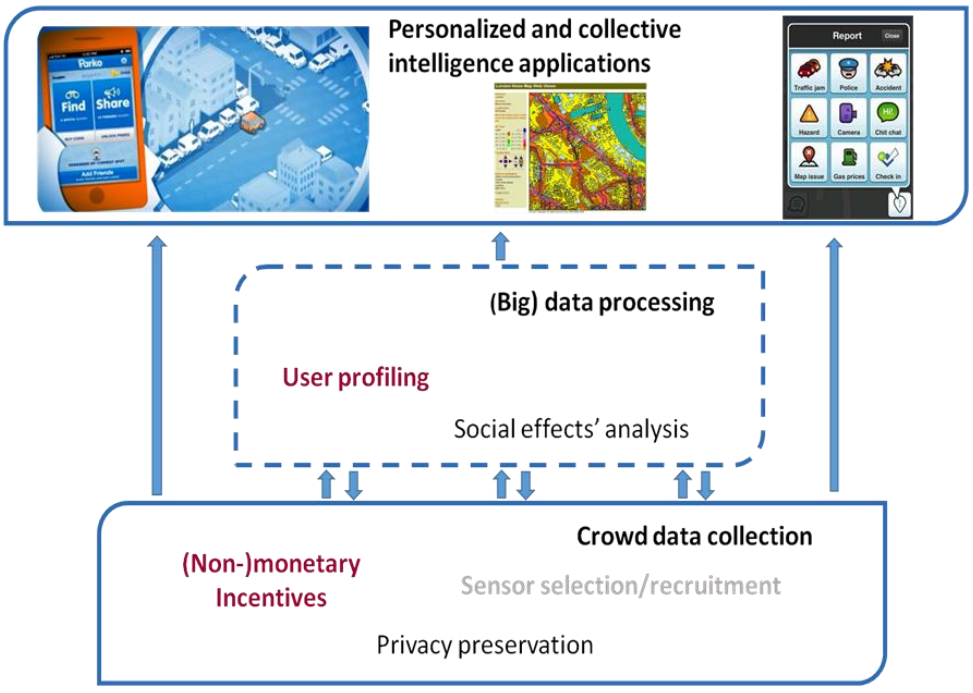


Figure 1.1 Typical anatomy of a participatory application

Both terms point to a fairly novel service provision paradigm, which aims to leverage the power of the crowds and the broad availability of smart mobile devices. It consists of collecting many small individual contributions from many users and building services out of them. As such, participatory sensing relies to high extent on the participation of the

crowds (users). More often than not, these end users need proper incentives to make contributions, whether involving monetary compensation or non-monetary ones. The right incentives need to take into account their individual interests, preferences, and behavioral traits.

1.2 User profiling and incentive provision

Generally, the term user profiling refers to the collection of data about the users and their characterization with regard to their behaviors, preferences and practices when using an application. For participatory sensing applications, in particular, the aim is mainly to infer the way different users make choices about whether to contribute to a task or not and which one to contribute to. Hence, we need to infer which attributes of tasks they assign most value to (e.g., the rewards they offer, their distance from the user location, their social or commercial mission). Likewise, part of this task is assessing their response to incentives of different kind:.

- **Monetary payments**: This mechanism actively motivates people to participate, since they might be not willing to perform certain activities without the proper award. The method of payment can be applied under specific rules, for example, regular payments, micropayments, coupons/credits [3][4].
- **Non-monetary payments:** Gamification is a technique used to motivate user to contribute. As users make contributions to different tasks, each task provides certain level of status. In gamification, user applications turn in game players, the competition among each other incentivizes the participatory sensing by earning respect for their contributions.

The focus in what follows is on participatory sensing applications that provide monetary rewards to end users as a form of incentive. For example, consider a participatory sensing platform that aims at organizing photo-shoots of coffee shops. The app might issue a task that entails recruiting users for getting photos from a specific coffee chain with, say, seven points of presence in a city. The owner of the coffee chain makes available some monetary budget for this purpose and the application platform undertakes to deliver the photo-shooting by making the best use of this budget, after accounting for its own reward. In other words, the platform should manage the available budget in the best possible manner, by directing the monetary rewards to those users who could take and contribute the best photos of the coffee shops. To this end, the platform needs to learn who the candidate

users are, who are the most skillful, as well as when increasing the offered reward really increases the chances of user participation and when not.

1.3 Outline of the dissertation

The aim of this dissertation is to study the individual user preferences in the particular context of the participatory sensing applications. The aim is to devise analytical models of these preferences that could be subsequently used for the systematic optimization of the budget allocated to them. Essentially, the dissertation work is split into three main phases, which proceed sequentially, *i.e.*, conclusion of the first phase is a prerequisite for the initiation of the next one.

1.3.1 Data collection

The first main task is to collect data regarding the preferences of the users, and the way they choose among different tasks available to them. In this project, this is pursued through online questionnaires, customly designed for the needs of this project. The related work is summarized in chapter 3 and includes the online survey tools that were tried and chosen for this purpose, the recruited subjects, and the pre-processing of the responses to the questionnaire. The outcome of this phase are the set of responses to the questionnaire per participant.

1.3.2 Data-driven user profiling

The data collected by the questionnaire's participants are then processed to profile them. The modeling draws on machine learning tools and was carried out using the MATLAB and Weka software. The work is outlined in chapter 4 and results in a profile for each questionnaire participant. However, the modeling assumptions/abstractions for the applications and the users are the subject of chapter 2.

1.3.3 Analysis of user profiles

In this last section, we explore the similarity/diversity that emerges out of the user profiles. We explore ways to cluster users and discuss the implications of the findings about the provision and proper targeting of incentives to them.

2 System model

In this chapter, we outline our basic assumptions about the way the participatory sensing application works and the way we model the choices end users make.

2.1 Users, tasks and user-to-task matching

Our focus is on applications used to simultaneously organize and carry out multiple tasks. Hence, at any point in time users log in the application, there is a list of ongoing tasks in the area around their location. The application takes into account the user profile and the physical location related to each task and decides to make offers to the user for a finite number, *K,* of them. The number *K* of offers is fixed, with K values anywhere in the range of [2, 10] being considered possible. Higher values would imply unrealistic cognitive load on the end user, also considering the screen sizes of the mobile devices.

Each task is described by a set of features. Such features may include the physical location of the task, corresponding to a different physical distance from each app user; the reward that the app offers for task contributions; the scope (e.g., commercial *vs.* community) of the task; and others. These features make up the task *attribute set a*

$$a = (\alpha_1, \alpha_2, \alpha_3, ...\alpha_M)$$

Users on the other hand, have individual preferences. For example, some of them attribute a lot of attention to the distance of a task, whether it is in their proximity or they need to walk/travel further away. Others are primarily motivated by monetary rewards, approaching their involvement in PS applications as a way to make some money; or others, may be driven by more altruistic ideals and exclusively be contributing to non-commercial tasks. Therefore, tasks that match the user preferences are more attractive for them and it is more probable to see them executed by them.

2.2 Users' choices of tasks as classification problems

The user choice setting is approached as a classification problem and is modeled with the help of logistic regression models. In the next section, we summarize briefly some background on classification problems and the logistic regression framework.

2.2.1 Classification problem and models

In machine learning and data mining, the purpose of classification models is to categorize subjects into a finite number of classes. Subjects are described by attributes, which may be categorical or numerical (discrete or continuous) and the model essentially makes a prediction as to which class should the subject be categorized into. The subjects' classification may be deterministic or probabilistic. In the first case, the model assigns a subject to a class with certainty, whereas in the second case, the subject is assigned to each class with a certain probability expressing the relative level of certainty for this assignment.

For example, consider that we would like to predict whether the request of a specific bank customer for a loan will be satisfied or not from the bank. The subjects are the loan requests and the two classes they may end up in correspond to their approval and rejection, respectively. The prediction could be made with the help of a classification algorithm that would be *trained* with the help of historical data, containing two types of information: the request attributes that are actually the requestor's attributes such as age, gender, employment status, saving, incomings; and the outcome of his/her request, accept or reject. An example of a classification model that would emerge could be as follows:

IF (Age in [30, 40] AND Income > 1000EUR AND Employment Status= Yes) OR (Age 20-29 AND gender= female AND income > 800)

THEN Loan Approval by the Bank

The classification model in this case consists of a set of rules and deterministically concludes whether the load request of a given customer will be accepted or rejected. The

threshold of the rules results from the training process, which may generally be supervised or unsupervised.

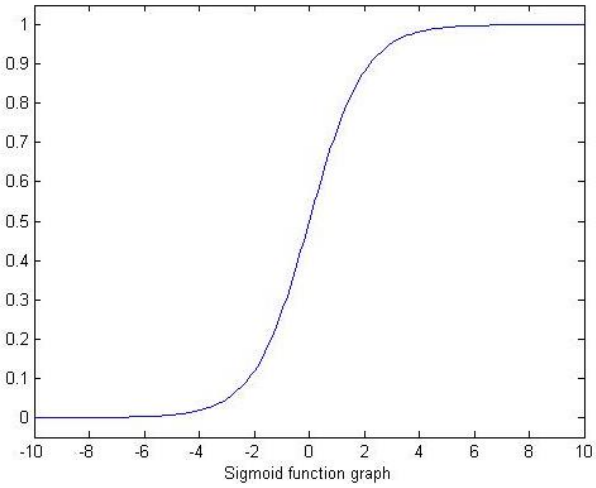


Figure 2.1 Plot of the sigmoid function

2.2.1.1 Supervised Learning

The training data (or labeled data) are prior subjects' data accompanied by labels that come from observation, measurements, logs etc. The labels report with certainty the outcome of the classification of past samples. In the previous example, the labels are the outcomes (approval or rejection) of past loan requests. Supervised learning is applied to the training of logistic regression models.

2.2.1.2 Unsupervised Learning

In unsupervised learning, the training data does not come with labels. Prediction is still possible based on the subjects' attribute description. Unsupervised learning is applied to clustering and dimensionality reduction.

2.2.1.3 The logistic regression framework

(Binary) logistic regression

This is one particular family of models for classifying subjects under multiple classes, *C*. For C=2, the model is called binary logistic regression model and predicts that a subject with feature vector x=(x_1, x_2, x_3,...,x_M) will be classified to class C_0 with probability

$$P(C_0 \mid X) = \frac{1}{1 + e^{-w_T x}} = \sigma(w_T x)$$

where w^T is the transpose of thefeature weight vector weighing the relative importance of each feature x_i upon the overall outcome. This equation is known as sigmoid function or logistic function [8].

As shown in Figure 2.1, g(z) tends towards 0 as z → -∞, and g(z) tends to 1 as z → ∞, hence $P(C_0|x)$ has probability semantics.

Multiclass logistic regression

There are several ways to generalize the binary logistic regression model into the case of multiple classes, *C* > 2. In this dissertation, we are concerned with two of them:

Multinomial logistic regression: The model computes probabilities of assigning a subject to each of the *C* classes as:

$$P(C_k \mid x) = \frac{e^{-w_T^k x}}{\sum_{j=1}^{C} e^{-w_T^j x}}$$

where now $w^k{}_T$ is the class-specific feature weight vector.

One *vs.* all approach: To solve a classification problem with C > 2 classes, we apply C independent binary logistic regression classifiers $P(C_k|x)$, k=1,2..*K*. Each one of them separates the feature values' space into two areas through *K* discriminant functions $P(C_k|x)$, k=1,2,3.

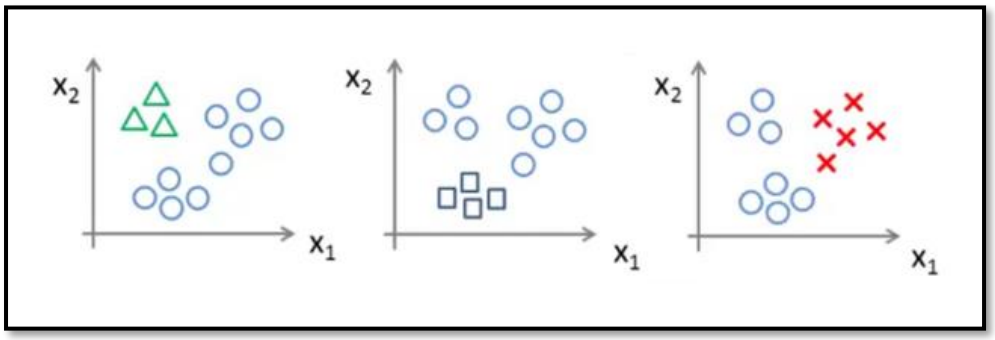


Figure 2.2 One vs all approach.

In Figure 2.2, we apply for each class a binary classification. As a result, three discriminant functions emerge with their respective weights. Each equation determines the probability of being triangle, square and x, respectively. To make a prediction for a new input x, the class that maximizes the probability

$$\max_k P(C_k|x)$$

is picked.

2.2.1.4 Model training and testing

The logistic regression model and its extensions for multiple classes have parameters (the weight vectors w^T) that need to be inferred (aka learned). This is carried out with the help of a training dataset and supervised learning techniques.

In the case of the multinomial regression model, for *C>2*, the training data set D contains the labels (assigned classes) t_i assigned to i past classification problem instances under feature vector values X_i. The labels t_i are 1-of-K vectors, that is vectors with one element equal to one (corresponding to the assigned class) and all the rest equal to zero.

$$D = \{ (X_1, t_1), (X_2, t_2), (X_3, t_3), \dots, (X_i, t_i) \} \tag{2.1}$$

One popular way to determine the weight parameters w^T is through maximum likelihood. The found weight parameters values maximize the posterior probabilities associated with the training data.

$$P(t|w_1..w_k) = \prod_{n=1}^{i} \prod_{k=1}^{C} P(C_k|X_n)^{t_{nk}}$$

$$= \prod_{n=1}^{i} \prod_{k=1}^{\mathrm{C}} y_{nk}^{t_{nk}}$$

where $y_{nk} = P(C_k|X_n)$. To ease computations, the standard practice is to consider the maximization of the logarithm of $P(t|w_1..w_k)$. Further, to avoid over-fitting phenomena that result in inaccurate predictions in real applications, it is necessary to add a regularization term [17]. In what is called ridge regression, as regularization penalty we add a quadratic term equal to the sum of weights, $\frac{\lambda}{2}||\mathrm{w}||^2$ (refer to Appendix for more details).

Hence, in the training process, we seek the weights that minimize what is called the cross-entropy error function

$$E(w_1 \dots w_k) = -\ln \mathrm{P}(\mathrm{t}|w_1..w_k) + \frac{\lambda}{2}|\mathrm{w}|^2$$

$$= -\sum_{n=1}^{i} \sum_{k=1}^{K} t_{nk} ln y_{nk} + \frac{\lambda}{2}|\mathrm{w}|^2$$

The derivative of the cross-entropy function is given by:

$$\nabla E(w_1, \dots, w_k) = \sum_{n=1}^{i} (y_n - t_n) X_n + \lambda w$$

This is a concave function [10, 17]; hence, it has a unique minimum that can be found if we apply the gradient descent algorithm and to efficiently iterate to the optimum value of the vector w, we use,

$$(w_1 \dots w_k)^{n+1} = (w_1 \dots w_k)^n - h\,\nabla E(w_1, \dots, w_k)$$

Where *h* is the step of the descent algorithm, which introduces a tradeoff between fast convergence (high *h* values) and high precision (small *h* values).

2.2.2 Mapping our problem setting to the generic multiclass classification problem

When we map our setting to the generic multiclass logistic regression model, we could make the following remarks:

- The users of the participatory sensing application are the subjects that are classified into four classes;
- The four classes correspond to the types of task offers users tend to accept. Class 1 (C1) corresponds to the physically closest and least rewarding task, class 2 (C2) to the second closest, class 3 (C3) to the second furthest and second best rewarding task offer, and second worst rewarding task, and Class 4 (C4) corresponds to offers for the furthest away tasks that are at the same time paying most for a contribution. The classes are listed in Table 2.1.
- The feature set considered in the classification could generally include a combination of task and user attributes over all four tasks. In our experimentation that follows, we consider two features for each task included in the offer. The first one is the user-task physical distance, as this emerges out of the locations of the user and the task. The second feature is the reward that each task offers for contributions to it. Therefore, the feature set for the classification problem includes eight features (four tasks times two features per task), *i.e.*, in Eq. (2.1) the size of vector X_i is 8x1 (d_1,p_1,d_2,p_2,d_3,p_3,d_4,p_4) and t_i is an 1-of-4 vector. All features are continuous.

  Note that, at a given time, the PS application presents the user with its offers, the first feature is *fixed*, whereas the second (reward) can be *dynamically controlled* to serve specific purposes (*e.g.*, encourage contributions towards a specific task that is not well served so far).

- The training dataset includes prior choices of users when they were presented with offers from the PS applications. In Eq. (2.1), t_i takes one of the 1-of-4 vector values shown in Table 2.2.
- each user *u* is modeled as a feature weight vector w_u^T .

	User-task distance, d	**Task reward, p**
Class 1	Minimum d_1	Minimum p_1
Class 2	Medium low d_2	Medium low p_2
Class 3	Medium high d_3	Medium high p_3
Class 4	Maximum d_4	Maximum p_4

Table 2.1 Class definition in our setting

Class	**Output**
Class 1	1, 0, 0, 0
Class 2	0, 1, 0, 0
Class 3	0, 0, 1, 0
Class 4	0, 0, 0, 1

Table 2.2. Possible outputs t_i depending on the user choice

3 Data collection and processing

The multiclass logistic regression models need to be trained with the help of datasets (training datasets). A special purpose dataset has to be collected accounting for the settings faced by the users of participatory sensing applications. In our case, in line with section 2.2.2, we need user responses to offers of four tasks at a time, each one coming up with a different distance from the user and a different reward.

There are different methods to collect a large set of such responses; we chose the option of online survey. Online surveys present numerous advantages against paper surveys or interviews when the aim is to collect a large number of samples. With them, it is easy and more time- and cost-efficient to manage, collect and store data [12]. Moreover, it is much easier to distribute at large scale to potential users willing to fill it. Specific channels to reach these users are email, social networking sites such as Facebook, as well as special-purpose websites that enable target potential survey participants. With respect to processing the collected data, we save the costs related to printing the survey, thus being more environmentally friendly.

3.1 Selection of the online survey platform

There are different online platforms providing survey-related services. Almost all of them present a free version with few features available to attract people. Typically, they place constraints on the number of responses, time available online, accessibility to export the data and so on. To gain more flexibility with these attributes, the online platforms request a subscription.

The platform has to meet the following requirements to be able to reach the best samples of data:

- Unlimited questions;
- Unlimited respondents;
- The format should be multiple choice;
- Block multiple responses from the same respondent;

- Present each question independently to the user. The option to go back to the previous question should be disabled;
- Export data in some established format, e.g., .csv files;
- Capability to generate a link to share among survey participants

The way different survey platforms cover these requirements is summarized in Table 3.1 and discussed in more detail in what follows.

	Google forms	Kwik Surveys Free Plan	Monkey Survey Free Plan	ESurvey Creator Student Plan (Free)
Unlimited number of questions	✓	✓		✓
Unlimited Number of Participants	✓	✓		✓
Unlimited Number of answers	✓	✓		
Free support	✓		✓	✓
Enhance Security SSL/HTTPS	✓	✓	✓	✓
Embed logo				✓
Multiple choose questions	✓	✓		✓
Add Images	✓			✓
Add theme	✓	✓		✓
Optimized for Smartphones and Tablet	✓		✓	✓
Prevent Multiple Participations	✓ Gmail account	✓ cookies	✓ cookies	✓ cookies & IP domain
Export Data CSV	✓	✓		✓
Multiple Collaboration	✓	✓		
Disable Back Button			✓	✓

Table 3.1 Comparison among different online platform for survey.

3.1.1 Google forms

This is a well-known tool to collect and organize information, with an easy environment to create surveys for free. To create a new survey, one needs to log in from a Gmail account. Through such an account, we can avoid multiple responses from single person [13]. On the negative side, the tool provides no interface to disable the option to go back to the question.

3.1.2 Kwik surveys

KwikSurvey was designed specifically for creating surveys. People from different background with or without experience are able to easily create their own survey, using the well-structured and quick to learn environment. Nevertheless, there is no option to prevent respondents from navigating back and forth among different questions. Although multiple responses can be prevented through the use of cookies, users don't need to log in to their email accounts, before starting answering the survey. [14].

3.1.3 SurveyMonkey

SurveyMonkey possesses a variety of characteristics. It is well developed to work either at smartphones or computers. SurveyMonkey is suitable and used for various areas of research such as marketing, educational, and customer services. It provides added-value services in terms of higher standards of security and privacy. On the other hand, to use all these characteristics it is necessary to subscribe for a professional account at a monthly fee. [15]

3.1.4 eSurvey Creator

This is the fourth solution we tried. It has an optimized set of features to be used either over smartphones or computers. It can be personalized with logo and layers to make it more attractive and increase the response rate. Unlike the other tools, to prevent multiple responses from a single user, it is possible to block them through either IP domain filtering or cookies. Moreover, the survey can be configured to not allow respondents to navigate back to previous questions. Some of these features are not available in the free version; however, they have made available a special offer for students, making it the best platform to use.

Therefore, our decision was to implement the questionnaire in this platform [16]

3.2 Questionnaire structure and participants

The questionnaire invites its participants to consider a particular scenario. According to this scenario, they are walking in a central urban district during their leisure time when they get a notification on their smartphone from a crowdsensing application they have subscribed to. The application presents them with four locations and invites them to choose one of them to go to and take photo for some monetary reward. Using their smartphone's GPS, the application can compute the distance of each of the four proposed locations from their current position.

The questionnaire then presents 22 different choice settings (aka. questions) to the participant.

3.2.1 Choice setting

Each question recommends four tasks to the participant and invites him/her to choose one of them. The screen for one such question is shown in Figure 3.1

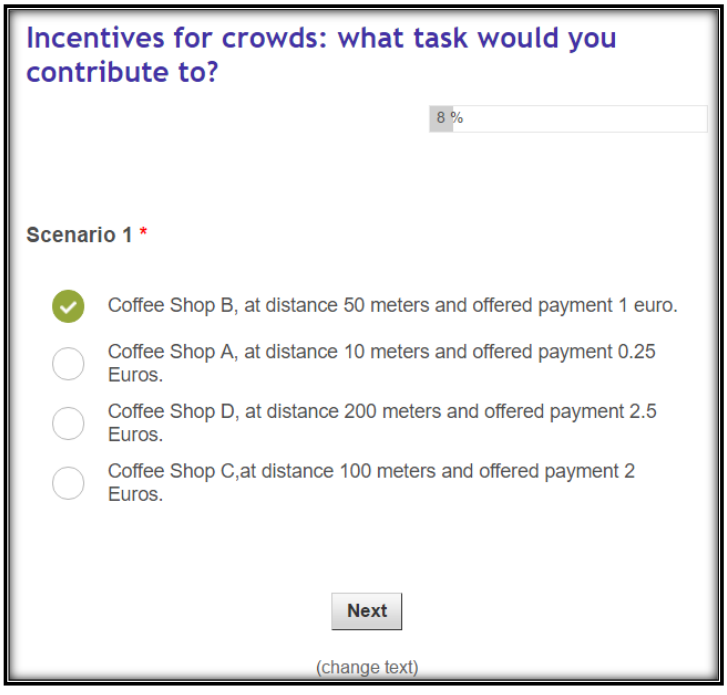


Figure 3.1 Scenario 1 sample

The task locations are four different coffee shops, each one related to a distance *d* and a payment *p*. The tasks in each question are chosen such that no task choice dominates the other three. In other words, the task choices always mark the same four classes:

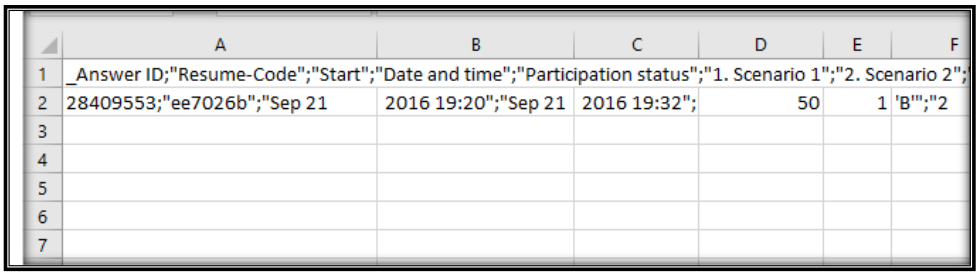
- class 1 (C_1): minimum distance and lowest payment;
- class 2 (C_2): second smallest distance and second lowest payment;
- class 3 (C_3): second largest distance and second largest payment; and
- class 4 (C_4): maximum distance and payment.

The distance of proposed tasks ranges in [0, 1000] meters and the proposed payments in [0, 6.50] EUR.

The respondents submit their preferences for each scenario, otherwise the platform does not let them advance to the next question. This ensures that there are no missing values by the time the responses are submitted.

3.2.2 Logging user choices

The answers collected by every respondent are recorded in the platform and can be exported in CVS format, which is compatible with Microsoft Office Excel and shown in Figure 3.2. It can also convert them to txt files.



	A	B	C	D	E	F
1	_Answer ID;"Resume-Code";"Start";"Date and time";"Participation status";"1. Scenario 1";"2. Scenario 2";					
2	28409553;"ee7026b";"Sep 21	2016 19:20";"Sep 21	2016 19:32";	50	1	'B'";"2
3						
4						
5						
6						
7						

Figure 3.2 Spreadsheet format.

The fields stored in the file are:

- Answer ID (anonymized), a unique identifier for a specific respondent
- Resume Code, an identifier that lets a respondent continue with the questionnaire from where he/she left it
- Start, the date and time the responses to the questionnaire were initiated
- Date and Time: the date and time of last response
- Participation status: general information to identify answers and acknowledge the status of each response

Then listed are the responses to each question. In Figure 3.1, the choice of the user at hand in question 1 is "B", which corresponds a 1 EUR payment ($p = 1\ EUR$) and distance 50 meters ($d = 50\ m$). The responses for the other 21 questions follow.

3.2.3 Questionnaire participants

Postgraduate students attending all Master programs running at the International Hellenic University were invited to answer this questionnaire. Additional people were attracted through personal contacts. Finally, responses were giving by advertised in crowdsourcing sites www.callforpaticpants.com and www.surveytandem.com.

3.3 Pre-processing of data

The stage of data pre-processing included two tasks:

3.3.1 Filtering out malicious/inconsistent/incomplete responses

We have taken some measures to prevent multiple responses from the same persons and filter out responses that may have been given either with malicious intentions or without much concentration.

For the first case, we have relied on functionality built in the online survey platform. The existence of such functionality was one of the platform choice criteria. For the second case, we inserted two questions (2 of the 22 in total), where one task choice dominated the other three, i.e., it offered the highest reward while lying closest to the user. These questions were placed as question no 10 and no 22. The intention was to also to see whether some user was fed up with answering all questions and after some point started answering randomly. Responders, who did not answer according to the intuition in these two questions, were excluded from the sample.

Finally, incomplete questionnaires were automatically discarded from the database.

Overall:

- 233 participants in total filled in the questionnaire;
- 61 participants filled in the questionnaire only partially;
- 40 participants were filtered out because of non-consistent replies to questions 10 and 20;

- 132 participants were retained and their replies were used as training datasets for the models developed in the rest of study.

3.3.2 Preparing data for input to MATLAB modeling routines

In a second pre-processing step, the residual data were transformed to a format capable to be used by MATLAB, the software that was primarily used for all our modeling scripts. To this end, the features of the proposed tasks were organized as a matrix X_u and the choices made by each user were presented in terms of a vector y_u.

$$\begin{matrix} p_{u1,1} & d_{u1,2} & p_{u1,3} & d_{u1,4} & p_{u1,5} & d_{u1,6} & p_{u1,7} & d_{u1,8} & y_{u1} \\ p_{u2,1} & d_{u2,2} & p_{u2,3} & d_{u2,4} & p_{u2,5} & d_{u2,6} & p_{u2,7} & d_{u2,8} & y_{u2} \\ & & & \vdots & & & & & \vdots \\ p_{u20,1} & d_{u20,2} & p_{u20,3} & d_{u20,4} & p_{u20,5} & d_{u20,6} & p_{u20,7} & d_{u20,8} & y_{u20} \end{matrix}$$

Figure 3.3 Matrix X_u and vector y_u

This specific format to be processed by using Matlab is easily implemented through the menu panel on eSurvey Creator. It suffices to manipulate the default options while exporting the collected data matrix X_u and vector y_u

The file is exported as a CVS file. The main requirement at this step is to delete unnecessary fields such as date and time, when the questionnaire was started and completed. Further, instead of having the information by arrays, containing successive responds, we change them to have them in columns.

At the end of this step, we end up with one matrix X_u and one vector y_u per user. The size of matrix X_u is 8 x 20 and that of vector y_u 1 x 20

3.4 Main processing of data in MATLAB

The main processing of the collected data and the derivation of user models is carried out in MATLAB. This processing roughly involves the following tasks:

3.4.1 Computing the feature weight vectors

This is carried out for both variants of the multiclass logistic regression models: the multinomial logistic regression (softmax function) and the one *vs.* all approach.

In the first case, we reuse the MATLAB built-in *mnrfit* routine. For the second case, we have written code drawing on the instructions of the Coursera online course "Machine Learning". The code is provided in Appendix B.

At the end of this step, each user is related to:

- one feature weight vector of size 27 for the multinomial regression model; and
- four feature weight vectors of size 36 for the one vs. all model variant

The variant of Softmax to have a weight vector of size 27 divided by three columns, where each column has 9 weight values. Each column represent one class versus the reference class. The first column contains the intercept values to predict the response of two classes, class A vs class D which is the reference class, class B vs class D and finally class C vs class D. In appendix C the weight vector can be presented by

- empty-matrix, in this case the weight vector is 0 for the 9 weight values in each column;
- in the case that only 9 weight values are presented means that in the other 2 columns the weight values are 0,
- the same way when if the weight values are 18 means that the last column the weight values are 0 and;
- finally, we have a vector of 27 which are the models which doesn't contain a complete column full of zeros.

3.4.2 Assessing the fitness of the two models

Two are the relevant measures of a model's accuracy:

- Training accuracy: showing how well the derived model fits the data that were used for the model training
- Testing or prediction accuracy: reflecting how well the derived model fits "new" data that were not part of the model training process

Ideally, we would need two different datasets for each user, one for training the models and the other for testing them. The common practice in case the two distinct datasets are not available is to use cross-validation techniques.

3.4.2.1 Cross-Validation

We have seen that, having a good training accuracy in our model from the data training does not mean that the results for predicting unseen data will be the same. Therefore, it requires the implementation of Cross-Validation. Cross-Validation is used as a statistical and machine learning technique to select the model which has the best accuracy on unseen data. [10, 22]. It consists of dividing the training data available into two segments, one of which will be used to build the predictor model and the second, to test the performance of the model (as a model validation). This method allows us to avoid overfitting problems, and helps to make a predictor model, which can be generalized to predict new data.

A well-known form of cross-validation is K-fold cross-validation. This consists in partitioning the training data into *K* partitions (of L=20/K samples each in our case). The K-1 partitions are used for training the model and the last partition is used for prediction purposes. One way to apply K-fold partition is by running it five times, each time

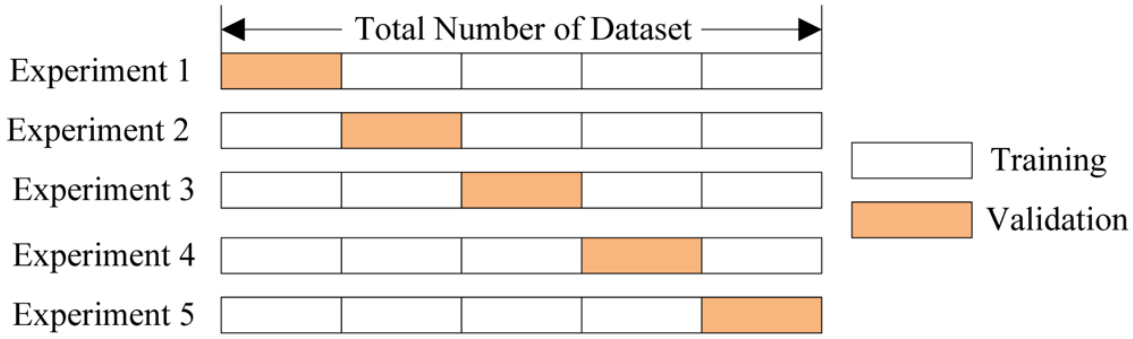


Figure 3.4 K-Fold Cross-Validation (with K=5)

changing which one of the K blocks serves as the prediction block. An alternative is to randomly generate the samples that will be used for prediction purposes. This way, one can sample the $\binom{20}{L}$ possible instantiations of the prediction block.

In the modeling work that is reported in Chapter 4, we have used K=4 (the rule-of-thumb is a split of 70-30 of the training data set). Therefore, the prediction set is of size L=-5.

We randomly generate 100 different prediction blocks and take their average as the model prediction accuracy score.

Computing the training accuracy of the models is more straightforward since all training samples are used.

4 Results

A total of 132 users emerged out of the questionnaire after the pre-processing of data. Every single user was processed to compute his/her weight feature vector according to the two approaches to multiclass logistic regression that are described in chapter 2: the one vs. all approach, where we essentially compute multiple binary logistic regression classifiers, and the multinomial regression model (softmax function).

In each case, we have computed:

- The feature weight vectors corresponding to each user;
- The resulting λ value used in the regularization process (ref. Appendix A);
- The training and prediction accuracy of the models, as an indication of how well these models capture the user preferences, as these are expressed in the questionnaires.

In Appendix C, we list the results for the first ten users, to give an idea of the kind of data that result in the modeling step. In what follows, we focus on the model accuracy aspects and the way different users compare with each other.

4.1 One vs. all implementation

Figure 4.1 and 4.2 show the histograms of test accuracy values in red colour and training accuracy values in blue colour, respectively. The histograms are computed over the 132 user samples, for which a model was built. The training accuracy performance in the majority of the models has to be controlled by varying the value of lambda. This reduces overfitting phenomena: namely, the training accuracy is reduced but, on the other hand, the test accuracy of the models increases and new unseen data can be predicted more accurately. For approximately 90% of the total number of samples, the training accuracy is higher than 60%; on the other hand, for approximately 61% of the total number of samples, the test accuracy is higher than 60%.

Figure 4.2 Training accuracy – one vs. all implementation

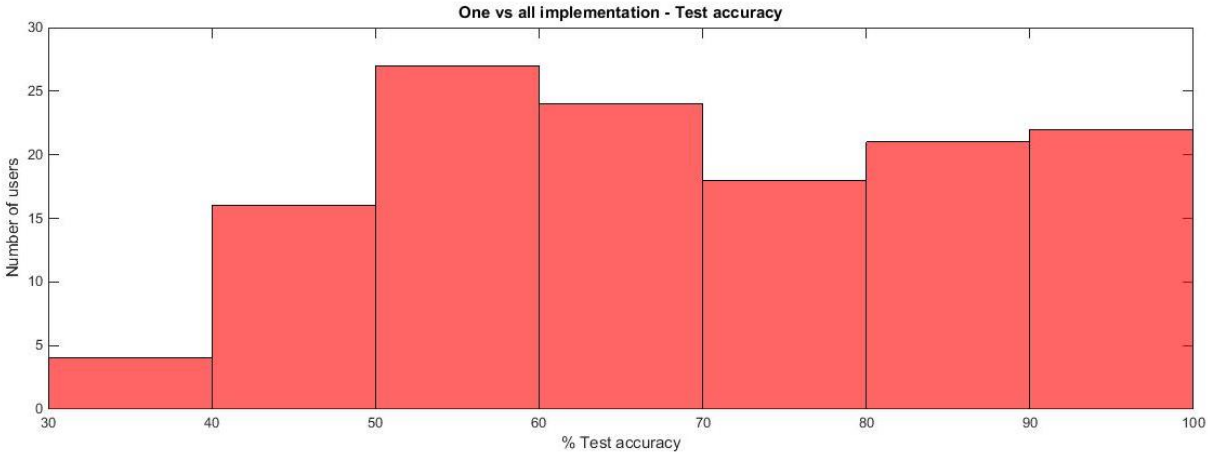


Figure 4.1 Test/ prediction accuracy – one vs. all implementation

4.2 Softmax implementation

Figures 4.3 and 4.4 plot the test accuracy in red colour and the training accuracy in blue colour. Plotted are teh histograms of the respective values, as computed out of the 132 user samples, for which the multinomial logistic regression model was built. Unlike in the one vs all implementation, in the implementation of Softmax we do not have a handle on the value of lambda to control the amount of regularization we apply to the algorithm. Hence, the training accuracy tends to be higher than we get when applying the one vs. all approach. For approximately 83% of the total number of samples, the training accuracy is 100%; on the other hand, the test accuracy is higher than 60% for approximately 67% of the total number of samples.

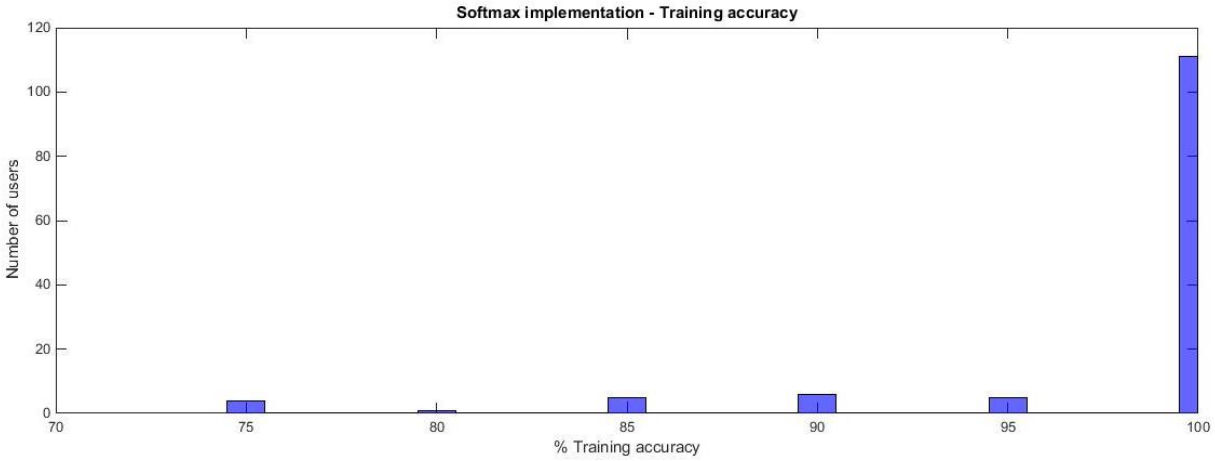


Figure 4.4 Training accuracy – Softmax implementation

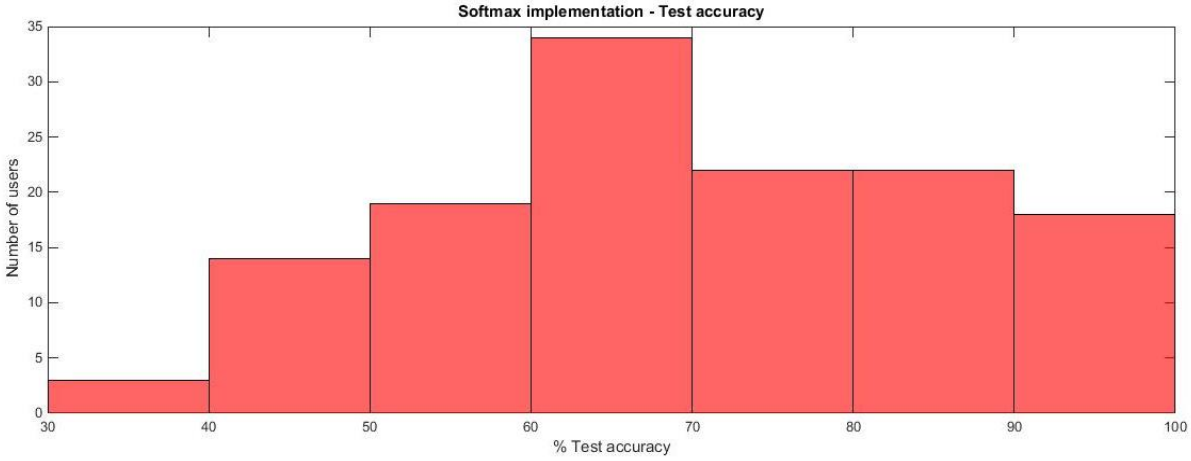


Figure 4.3 Test/ prediction accuracy – Softmax implementation

4.3 Training and prediction accuracy comparison

Figures 4.2 and 4.4 are the histograms of the training accuracy for the one vs all implementation and Softmax implementation, respectively. For approximately 83% of the total number of samples, their training accuracy is 100% accuracy. It presents higher accuracy comparing to Softmax implementation which it approximately provides 40% of the total number of samples in the range of 90% -100% training accuracy. This is expected because in one vs all we penalize the overfitting to maximize the model to have better results in the test accuracy.

Figures 4.1 and 4.3 are the histograms of the test accuracy for the one vs. all implementation and the Softmax implementation, respectively. In this case, the sofmax performance is slightly better than the one of the one. vs. all approach.

Overall, the best performance is achieved by the Softmax implementation, which outperforms its competitor approach in both the training and prediction scores. Moreover, if we also account for the time taken to build the model, Softmax could produce the model faster rather than one vs. all. In any case, this is a case study comparison and the results cannot be generalized although the one vs. all technique has an inherent disadvantage in that it ends up working with asymmetric training datasets, even if the original dataset is a symmetric one. Using alternative techniques and methods allows us to have better chances to get a good model solution.

4.4 Similarity/diversity of users

The final step is to compute the similarity/diversity of users. Besides the informational value of this exercise, it has practical implications giving us a clear sign as to the extent in which we could define classes of users rather than targeting each one independently (one class per user). Hierarchical clustering is a well-known technique in data mining to build a hierarchical diagram by grouping objects according to how similar they are. A cluster tree is built through this algorithm, which basically computes their similarity/diversity by measuring the distance of every pair of objects (user models). After, having the proximity values between each user in the data set, it can be determined how the users should be grouped into clusters [24]. However, the distance can be computed according to different metrics such as the Euclidean distance, squared Euclidean distance, City block metric, Minkowski distance, cosine similarity among others [25]. Herein, we rely on the most popular metric to compute distance, *e.g.*, the Euclidean distance. [26, 27].

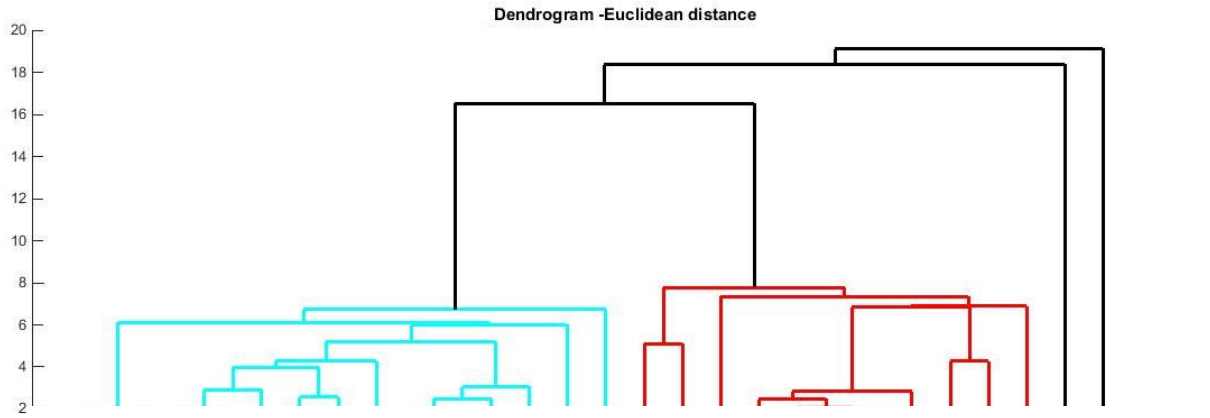


Figure 4.5 Dendrogram – Method single linkage

A dendrogram represents objects in a hierarchical tree and the height is proportional to the distance between two objects (user models) being connected by a link. Figure 4.6 presents the dendrogram that emerges when applying the single linkage method, where the distance between two clusters is taken to be the minimum one over all pairs of nodes belonging to different clusters. It is viewed starting from the top as a single cluster covering all the objects. Nevertheless, we can define a threshold to have the objects grouped into different clusters. In this particular analysis, if the threshold is set to 8, we can have users grouped in four classes. If we choose a lower value of threshold, more clusters come up but now the inter-cluster distance is smaller.

Figure 4.7 presents the clustering that emerges when we apply the complete linkage method, whereby the distance of two clusters is taken to be the highest distance between any two nodes belonging to different clusters. This time we come up with three clusters, when we use the default threshold value of 8. Unlike the method of single linkage, the complete linkage tends to discover clusters that are more equal in terms of diameters and

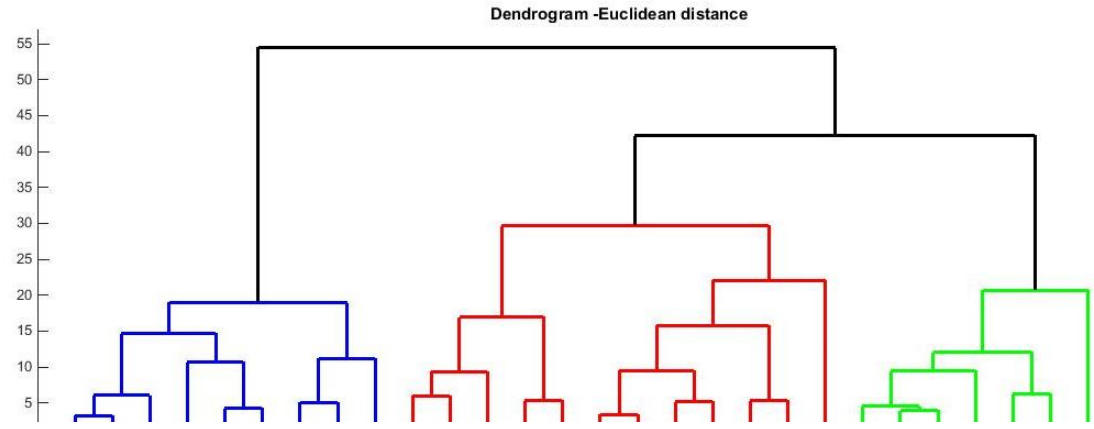


Figure 4.6 Dendrogram – Method complete linkage

size. Therefore, three classes of users are defined although in the same way to the previous method, it is up to us to define the best value of threshold, which optimize the definition of the classes. We can deduce from the analysis of the data the number of classes at least four classes of users and three classes of users applying single and complete linkage respectively.

Defining the value of threshold is completely arbitrary. In figure 4.7 its value was chosen to be 5.5. This particular value is lower than the default presented in figure 4.5 and shows clusters of users, which are closer in terms of their preferences. The single linkage method was used. It can be identified 5 clusters although it is in total 11 clusters. The nodes 13, 9, 10, 19 and 17 are not connected with others. This can be either because the method to compute the distance has a drawback trying to build long thin clusters that choose elements with the shortest distance; or, it can be the result of users whose behaviour is simply distinctly different than all the other users.

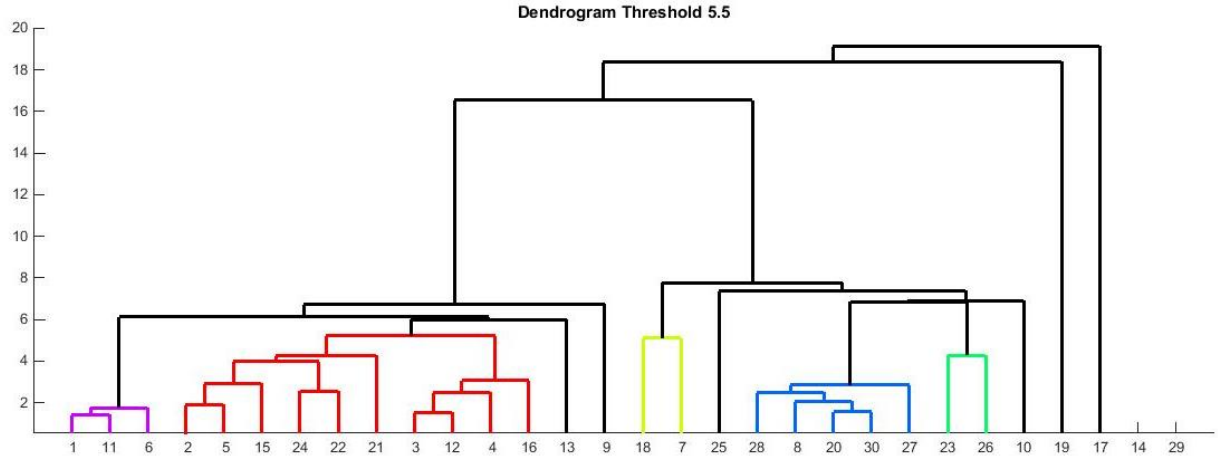


Figure 4.8 Dendrogram Threshold 5.5 -Method single linkage

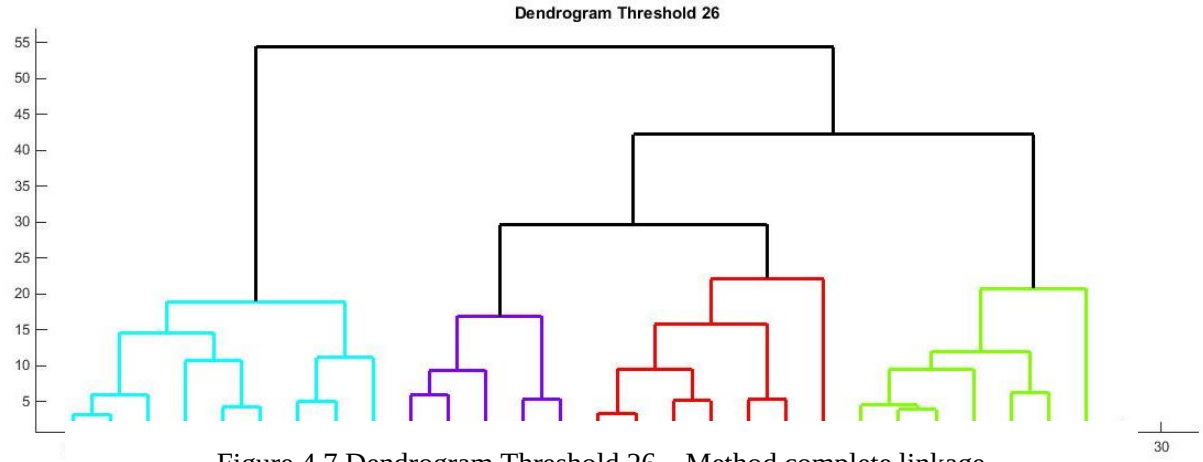


Figure 4.7 Dendrogram Threshold 26 – Method complete linkage

In figure 4.8, the cluster was built choosing the complete linkage method. Computing the distance through the complete linkage method leads to compact clusters with equal diameter. When the threshold is set to 26, the result is 4 clusters; in this case, no node remains unassigned to clusters. Therefore, four or five clusters can be a good representation of classes of users, while the distance among each other within the cluster is a good representation of users who share common preferences.

5 Conclusions and future work

This dissertation was written as a part of the MSc in ICT Systems at the International Hellenic University. The aim of this dissertation was the study that revolves around participatory sensing, building services through the small and individual contribution by many users. The goal of this project was to use past data which consist of twenty-two scenarios offering four tasks with different incentives, monetary reward and distance. Base on their choices, every single user was described as a feature weight vector applying machine learning techniques to iterate their similarity or diversity to possibly define classes of users instead of tracking individual's preferences for each user.

5.1 Conclusions

In participatory sensing applications users need to be provided with proper incentives for their participation in such applications. In order to optimize the budget in campaigns which requires the recruitment of users, doing so is necessary to learn the behavior, interests and preferences of the user. The penetration of smartphones and tablets capable of connecting to the internet with different sensors make an opportunistic way to build more sensing applications. Therefore, the importance of profiling users to better provision and proper targeting of incentives.

The dataset consists in 132 user participations and it was processed by Matlab. A great percentage of these samples came from students and alumni at IHU. Further, we posted the questionnaire in crowdsource special purpose websites such Call for applicants and SurveyTandem to reach as many samples as possible. The second phase of this project was to build the models for each users. The important elements in these models is the weight vector feature for every single user. Moreover, we use two different classifiers to compute multi-class logistic regression, one vs all and Softmax classifier, respectively. Both implementations had almost the same performance regarding the test accuracy unlike training accuracy which Softmax implementation had better performance. This is acceptable due to the one vs all implementation, we had the parameter lambda which it works as a regularization to avoid underffiting in the model. Further, we had the results of the test and training accuracy for each user.

The final process was to define classes of users who share same characteristics. we clustered the 132 weight vector features. Having as a results, users who give same importance either preferring short distances without caring about to receive higher payment or choosing the one that pay most and finally some users who compromise distance with payment. Although, the sample covered a small portion of the population, the approach is well generalized to identify classes of users to be used in crowdsensing applications.

5.2 Future work

The presented research covers models of users using real data collected through the participation of 132 people which has previously been processed to delete those malicious responses which can generate noise in the models presented. Therefore, these models are suitable to be used as inputs to an optimization problem for user-task assignment and payment allocation to maximize the quality of user contributions as extension to the Mobihoc ’16 work[1]

[1] First learn then earn: optimizing mobile crowdsensing campaigns through data-driven user profiling, ACM Mobihoc ‘16, Paderborn, Germany

Bibliography

[1] Francesco Restuccia, Sajal K. Das, Jamie Payton. 2015. Incentive Mechanisms for Participatory Sensing: Survey and Research Challenges ACM Trans. Sensor Netw. V, N, Article A (January YYYY), 38 pages.

[2] Luo, T. and Tham, C-K. Fairness and Social Welfare in Incentivizing Participatory Sensing. Annual IEEE Communication Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2012.

[3] S. Reddy, D. Estrin, M. Hansen, and M. Srivastava, "Examining micropayments for participatory sensing data collections," in Proceedings of the 12th ACM international conference on Ubiquitous.

[4] The Medicare Access & Chip Reauthorization Act of 2015.The Merit-Based Incentive Payment System: MIPS Scoring Methodology Overview.

[5] Voznika, F. & Viana, L. Data Mining Classification. [Online].[Accessed 25 August 2016]. Available from: https://courses.cs.washington.edu/courses/csep521/07wi/prj/leonardo_fabricio.pdf

[6] Cawley, G. Talbot, N. Girolami, M. Sparse Multinomial Logistic Regression via Baseyian L1 Regularisation. [Online]. Available from: https://papers.nips.cc/paper/3155-sparse-multinomial-logistic-regression-via-bayesian-l1-regularisation.pdf [Accessed 25 August 2016].

[7] Mohri, M. Rostamizadeh, A. & Talwalkar, A. Foundations of Machine Learning (Adaptive computation and machine learning series). Cambridge, MA, USA: The MIT Press, 2012.

[8] AY Ng, Machine Learning (Course handouts). CS229. Standford University, [Online] Available from: cs229.stanford.edu/notes/cs229-notes1.pdf [Accessed 25th August 2016].

[9] Murphy Keving P. Machine Learning: A Probabilistic Perspective. University of British Columbia, Canada. 2011. [Online] Available from: http://www.cs.ubc.ca/~murphyk/MLbook/pml-intro-5nov11.pdf [Accessed 15th October 2016].

[10] C. M. Bishop. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Secaucus, NJ, USA, 2006.

[11] UC Irvine Machine Learning Repository [Online], Available from: https://archive.ics.uci.edu/ml/index.html [Accessed 15th October 2016].

[12] Chenicheri Sid Nair & Phillip Adams (2009) Survey Platform: A Factor Influencing Online Survey Delivery and Response Rate, Quality in Higher Education, 15:3, 291-296

[13] Google Forms – create and analyse surveys, for free. [Online] Available from: https://docs.google.com/forms/u/0/ [Accessed 15th October 2016].

[14] KwikSurveys: Free online survey & questionnaire tool. [Online] Available from: https://kwiksurveys.com/ [Accessed 15th October 2016].

[15] SurveyMonkey: Free online survey sorftware & questionnaire tool. [Online] Available from: https://www.surveymonkey.com/ [Accessed 15th October 2016].

[16] Create Free Online Surveys & Questionnaire with eSurvey Creator. [Online] Available from: https://www.esurveycreator.com/ [Accessed 15th October 2016].

[17] Karaliopoulus M., Koutsopoulos I. & Titsias M. First Learn then Earn: Optimizing Mobile Crowdsensing Campaigns through Data-driven User Profiling. Department of Informatics, Athens University of Economics and Business. Athens, Greece.

[18] Browlee J. Gradient Descent for Machine Learning. 2016 [Online] Available from: http://machinelearningmastery.com/gradient-descent-for-machine-learning/ [Accessed 15th October 2016].

[19] Brownlee J. Overfitting and Underffiting with Machine Learning Algorithms. 2016 [Online] Available from: http://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/ [Accessed 15th October 2016].

[20] Domingos P. A Few Things to Know about Machine Learning. Washington. U.S.A. [Online] Available from: https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf pdf [Accessed 15th October 2016].

[21] Ng A. Y. Feature selection, L1 vs. L2 regularization, and rotational invariance. Computer Science Department, Stanford University. California. USA. [Online] Available from: http://www.machinelearning.org/proceedings/icml2004/papers/354.pdf [Accessed 15th October 2016].

[22] Refaelilazdeh P., Tang L. & Liu H. Cross-Validation. [Online] Available from: http://leitang.net/papers/ency-cross-validation.pdf [Accessed 15th October 2016].

[23] Ng A. Y. Advice for Applying Machine Learning. Stanford University (Course handouts). CS229. [Online] Available from: http://cs229.stanford.edu/materials/ML-advice.pdf [Accessed 15th October 2016].

[24] https://es.mathworks.com/help/stats/hierarchical-clustering.html

[25] https://es.mathworks.com/help/stats/clusterdata.html

[26] http://www.statsoft.com/Textbook/Cluster-Analysis#h

[27] Yim O. & Ramdeen K.T. Hierarchical Cluster Analysis: Comparison of Three Linkage Measures and Application to Pshycological Data. 2015.[Online] Available from: http://www.tqmp.org/RegularArticles/vol11-1/p008/p008.pdf [Accessed 22nd December 2016].

Appendix A

A1 Overfitting and underfitting

In machine learning, the training data set is used to build a predictor model. Containing a limited number of samples with random noise, the task is to fit a model to describe the underlying relationship. Overfitting produces good results in describing the samples of the training set and the error function tends to zero; but, on the other hand, it is less accurate in describing unknown data samples [11, 19, 20].

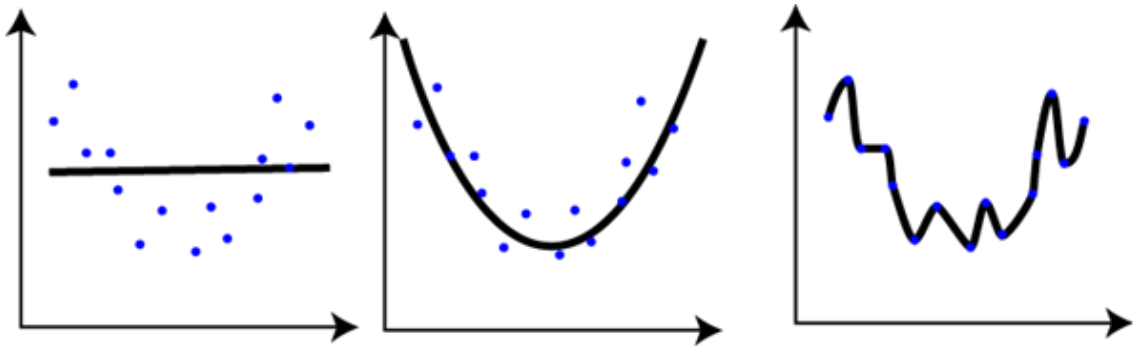


Figure A. 1 Overfitting demonstration

Figure A. 1 presents three graphs. The first one, which is a linear model, does not describe correctly the data. Increasing the order of the polynomial, we can fit the trend in the sample data to adequate accuracy. The third graph presents overfitting, the higher-order polynomial fits exactly each data point. To do so, it typically presents many oscillations and high weight values that let it change value quickly from sample to sample. Nevertheless, the model does not generalize well when predicting new values because, in its attempt to fit tightly the training dataset, it also fits noise that distorts the real underlying data structure. Therefore, the model plotted in graph two, may result in smaller error because it fits the data trend without being firmly attached to the possible noise of the training dataset.

On the other hand, underfitting refers to a model, which does not describe well the training dataset, neither can it generalize to predict new data [19]. The first graph describes

perfectly this phenomenon in Figure A. 1, being the result of an excessively simplistic model.

A2 Regularization

Overfitting is one of the reasons behind low accuracy in predictions. Typically, overfitting is accompanied by large values of the feature weight vector w_u^T elements. Regularization is a technique responding to the overfitting threat by adding to the error cost function an additional cost term penalizing large weight values [10, 21]

$$E(w_{u1}..w_{uk}) = -\sum_{n=1}^{i}\sum_{k=1}^{K} t_{nk} ln y_{nk} + \frac{\lambda}{2}||w_u||^2$$

the term $\frac{\lambda}{2}||w_u||^2$ is the regularized parameter, where λ is the term which controls the importance of the regularization compared to the error function. Evidencing overfitting and large coefficient values in the model, sets a clear case for applying regularization. The value of λ is typically found by trial and error, which consists of repeatedly running the training algorithm, using different values of $\lambda \geq 0$ and finally choosing the value that yields the best results.

A3 Bias / Variance tradeoff

High bias and high variance are both sources of error in supervised learning algorithms. [10 23] These lead to building a prediction model, which does not generalize to predict unseen data. Further, the diagnostic of one of these or both might be key to use mechanisms to have much better results.

- High bias is presented when the prediction model has an unacceptable high error and the training error as well. Further, the gap among each other is a minimum value. For linear regression, having a complex model which we use a simple model to fit it is called “high bias”. In others words, it is presented when we have underfitting in our model.

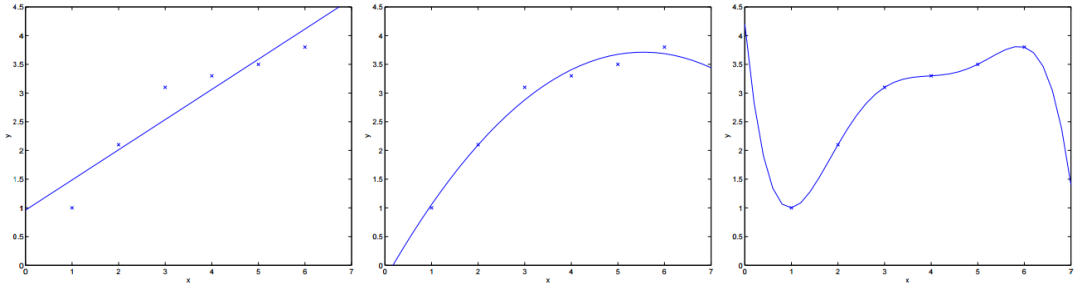


Figure A. 2 Bias / Variance tradeoff

- On the other hand, High variance is presented when the error on the training set is a low value and the error to generalize new unseen data trend to be really high. In this case, the model to generalize the unseen data is too complex which suffer from overfitting.

Figure A. 2 presents three curves. The three curves aim to fit the N points samples. The left one is the simplest model, a linear model of the form $y = w_1 x + w_0$. On the other hand, the right model is more complex using a higher polynomial to fit the points data. The polynomial in the form $y = \theta x^5 + \cdots + \theta_4 x + \theta_5$. Both models are not well generalizing to predict new unseen data. The leftmost presents high bias, a simple model which it is observed underffiting and the rightmost high variance which the model is overfitting and present too complex model.

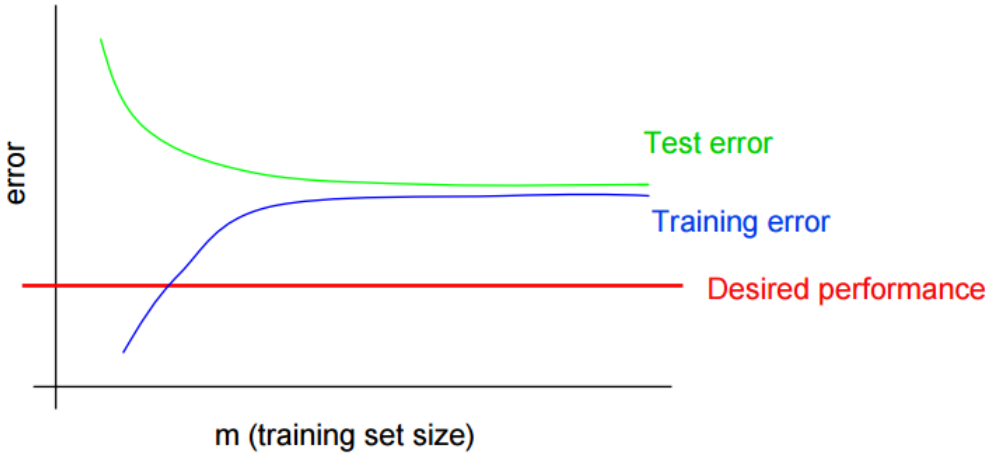
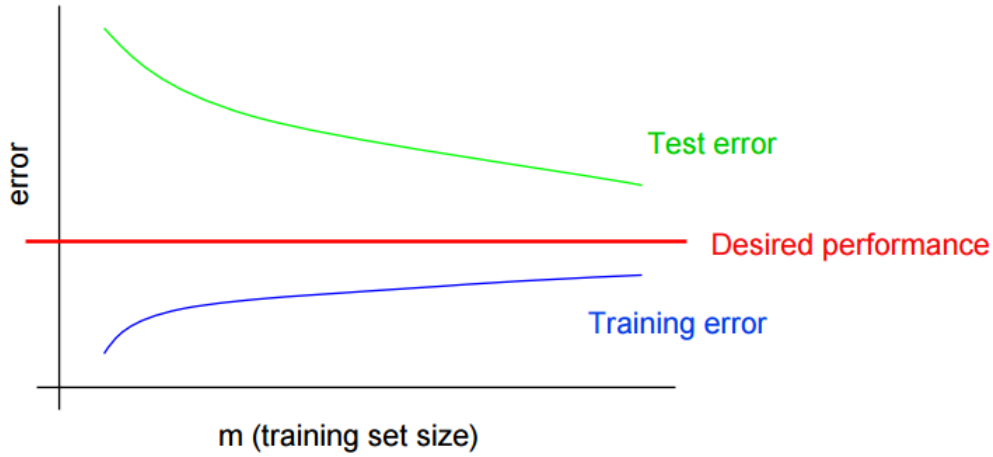


Figure A. 3 Typical curve for High Bias

The curve for high bias can be diagnosed by plotting the curve error vs m, where m is the size of the training set. It results in high error performance according to the desire one. Both test error and training error.

The curve for high variance can be diagnosed by plotting the curve error vs m, where m is the size of the training set. It results in high error performance according to the desire



one. Unlike, high bias which both training and test error are above to the desired performance. High variance oresents an acceptable training error.

Through the diagnostic of one of this phenomenon. It can lead us what we can try next to solve it.

Figure A. 4 Typical curve for High Variance

DIAGNOSTIC	WHAT TO DO NEXT?
HIGH BIAS	Increase the number of set features in our model.
HIGH BIAS	Increase the order of the polynomial features.
HIGH BIAS	Reduce the value of lambda.
HIGH VARIANCE	Use more training data examples.

HIGH VARIANCE	Reduce the number of set features in our model.
HIGH VARIANCE	Increase the value of lambda.

Table A. 1 Recommended actions when high bias/variance is evidenced in the model

Appendix B

B.1 MATLAB scripts for modeling users

-- Main script for the user modelling with the softmax function

```
%% Initialization
clear; close all; clc
load user3
K=4;

indices=crossvalind('Kfold',y,K);
for i_fold=1:K
    test = (indices == i_fold);
    train = ~test;
    [B,dev,stats]=mnrfit(X(train,:),categorical(y(train,:)))
    p=mnrval(B,X(test,:))
    [p_max, i_max]=max(p, [], 2)
    p = i_max;
    selection(i_fold,:)=mean(double(p == y(test,:)) * 100)
    mean(selection)
    fprintf('\nTraining Set Accuracy: %f\n', mean(double(p == y(test,:)) * 100))
end
```

-- Main script for the user modelling with the one-vs-all function

```
clc,clear all

load user1
K=4;
num_labels =8 ;
indices=crossvalind('Kfold',y,K);
for i_fold=1:K
    test = (indices == i_fold);
    train = ~test;
    m = size(X(train,:), 1);
    fprintf('\nTraining One-vs-All Logistic Regression...\n')
    lambda =0.3672;
    [all_theta] = oneVsAll(X(train,:), y(train,:), num_labels, lambda);
    fprintf('\nTraining One-vs-All Logistic Regression...\n')
    pred = predictOneVsAll(all_theta, X(test,:));
    selection(i_fold,:)=mean(double(pred == y(test,:)) * 100)
```

```
    mean(selection)
    fprintf('\nTraining Set Accuracy  : %f\n', mean(double(pred == y(test,:))) * 100);
end
```

-- Function One vs all

```
function [all_theta] = oneVsAll(X, y, num_labels, lambda);
m = size(X, 1);
n = size(X, 2);

all_theta = zeros(num_labels, n+1);
X = [ones(m, 1) X];

for i = 1:num_labels;
    initial_theta = zeros(n+1, 1);
    options = optimset('GradObj', 'on', 'MaxIter', 100);
    [theta] = fmincg(@(t)(lrCostFunction(t, X, (y==i), lambda)), initial_theta, options);
    all_theta(i, :) = theta';
end

End % function
```

-- Function lrCostFunction

```
function [J, grad] = lrCostFunction(theta, X, y, lambda)

m = length(y);
J = 0;
grad = zeros(size(theta));
h = sigmoid(X*theta);

theta1 = [0 ; theta(2:end, :)];
p = lambda*(theta1'*theta1)/(2*m);
J = ((-y)'*log(h) - (1-y)'*log(1-h))/m + p;
grad = (X'*(h - y)+lambda*theta1)/m;
end
```

-- Function Sigmoid

```
function g = sigmoid(z)
%SIGMOID Compute sigmoid functoon
%   J = SIGMOID(z) computes the sigmoid of z.
g = 1.0 ./ (1.0 + exp(-z));
```

```
end
```

-- Function fminc

```
function [X, fX, i] = fmincg(f, X, options, P1, P2, P3, P4, P5);

% Read options
if exist('options', 'var') && ~isempty(options) && isfield(options, 'MaxIter');
    length = options.MaxIter;
else
    length = 100;
end

RHO = 0.01; SIG = 0.5; INT = 0.1; EXT = 3.0; MAX = 20; RATIO = 100;

argstr = ['feval(f, X'];
for i = 1:(nargin - 3);
    argstr = [argstr, ',P', int2str(i)];
end
argstr = [argstr, ')'];

if max(size(length)) == 2, red=length(2); length=length(1); else red=1; end
    S=['Iteration '];

    i = 0;
    ls_failed = 0;
    fX = [];
    [f1 df1] = eval(argstr);
    i = i + (length<0);
    s = -df1;
    d1 = -s'*s;
    z1 = red/(1-d1);

    while i < abs(length);
        i = i + (length>0);

        X0 = X; f0 = f1; df0 = df1;
        X = X + z1*s;
        [f2 df2] = eval(argstr);
        i = i + (length<0);
        d2 = df2'*s;
        f3 = f1; d3 = d1; z3 = -z1;
        if length>0, M = MAX; else M = min(MAX, -length-i); end
        success = 0; limit = -1;
        while 1;
         while ((f2 > f1+z1*RHO*d1) | (d2 > -SIG*d1)) & (M > 0) ;
            limit = z1;
            if f2 > f1;
                z2 = z3 - (0.5*d3*z3*z3)/(d3*z3+f2-f3);
            else
```

```
                A = 6*(f2-f3)/z3+3*(d2+d3);
                B = 3*(f3-f2)-z3*(d3+2*d2);
                z2 = (sqrt(B*B-A*d2*z3*z3)-B)/A;
            end
            if isnan(z2) | isinf(z2);
                z2 = z3/2;
            end
            z2 = max(min(z2, INT*z3),(1-INT)*z3);
            z1 = z1 + z2;
            X = X + z2*s;
            [f2 df2] = eval(argstr);
            M = M - 1; i = i + (length<0);
            d2 = df2'*s;
            z3 = z3-z2;
        end
        if f2 > f1+z1*RHO*d1 | d2 > -SIG*d1
            break;
        elseif d2 > SIG*d1
            success = 1; break;
        elseif M == 0
            break;
        end
        A = 6*(f2-f3)/z3+3*(d2+d3);
        B = 3*(f3-f2)-z3*(d3+2*d2);
        z2 = -d2*z3*z3/(B+sqrt(B*B-A*d2*z3*z3));
        if ~isreal(z2) | isnan(z2) | isinf(z2) | z2 < 0
            if limit < -0.5
                z2 = z1 * (EXT-1);
            else
                z2 = (limit-z1)/2;
            end
        elseif (limit > -0.5) & (z2+z1 > limit)
            z2 = (limit-z1)/2;
        elseif (limit < -0.5) & (z2+z1 > z1*EXT)
            z2 = z1*(EXT-1.0);
        elseif z2 < -z3*INT
            z2 = -z3*INT;
        elseif (limit > -0.5) & (z2 < (limit-z1)*(1.0-INT))
            z2 = (limit-z1)*(1.0-INT);
        end
        f3 = f2; d3 = d2; z3 = -z2;
        z1 = z1 + z2; X = X + z2*s;
        [f2 df2] = eval(argstr);
        M = M - 1; i = i + (length<0);
        d2 = df2'*s;
    end

    if success
        f1 = f2; fX = [fX' f1]';
        fprintf('%s %4i | Cost: %4.6e\r', S, i, f1);
        s = (df2'*df2-df1'*df2)/(df1'*df1)*s - df2;
        tmp = df1; df1 = df2; df2 = tmp;
```

```
        d2 = df1'*s;
        if d2 > 0
            s = -df1;
            d2 = -s'*s;
        end
        z1 = z1 * min(RATIO, d1/(d2-realmin));
        d1 = d2;
        ls_failed = 0;
    else
        X = X0; f1 = f0; df1 = df0;
        if ls_failed | i > abs(length)
            break;
        end
        tmp = df1; df1 = df2; df2 = tmp;
        s = -df1;
        d1 = -s'*s;
        z1 = 1/(1-d1);
        ls_failed = 1;
    end
end
```

--Function Predict One vs. All

```
function p = predictOneVsAll(all_theta, X);

m = size(X, 1);
num_labels = size(all_theta, 1);

p = zeros(size(X, 1), 1);
X = [ones(m, 1) X];

ps = sigmoid(X*all_theta');
[p_max, i_max]=max(ps, [], 2);
p = i_max;
end
```

APPENDIX C

Below we show indicatively the modelling parameters, as estimated for ten users.

The matrix lists the per-user weights of the softmax function (column 6) and those of the 4 one-vs.-all binary classifiers (column 2). It also reports the optimal lambda value that resulted from the cross-validation process and the resulting accuracies of the two variants of multiclass classification.

User id	**One vs all**	**Training accuracy**	**Test accuracy**	**λ**	**Softmax**	**Training accuracy**	**Test accuracy**
29698744	-0.0725 -0.0217 -0.0740 0.0229 -0.0873 -0.0062 -0.0912 0.0024 -0.0972 -0.0007 0.2932 0.0004 -0.2974 -0.0013 0.0003 -0.0012 0.0284 0.0012 0.0640 0.0248 0.0689 0.5101 0.0692 -0.6728 0.0865 0.1485 0.0585 -19.8883 -0.0000 -0.0000 -0.0002 -0.0000 -0.0003 -0.0000 -0.0004 -0.0000	95%	80.83%	0.05	-218.81, -0.797 142.987, -0.137 39.059, -0.172 24.364, 0.042 -5.957	100%	80.42%
29699850	-0.0013 0.0091 -0.0015 -0.0082 -0.0019 0.0040 -0.0024 -0.0039 -0.0055 0.0079 -0.0090 0.0090 0.0080 0.0111 -0.0041 0.0144 0.0037 0.0334 -22.3543 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -22.3543 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000	90%	85.83%	0.5	148.878, 0.478 -87.337, -0.183 28.574, 0.180 -24.948, 0.248 -53.678	100%	85.83%

29700049	0.0235 0.0671 0.0170 0.3369 0.0178 -0.4422 0.0191 0.0912 0.0165 -0.0082 -0.0036 -0.0108 0.0079 -0.0089 -0.0077 -0.0131 0.0008 -0.0132 -0.2190 0.0041 -0.2426 -0.0095 -0.1299 0.0087 -0.2022 -0.0000 -0.1959 0.0069 -0.0390 0.0124 0.0021 0.0015 -0.0074 0.0130 0.0085 0.0016	70%	54.17%	1	41.370, 168.79 -77.685, 0.3 0.808, 0.19 -39.061, -148.855 24.415,0.08 0.072, -0.161 -22.277, -0.889 26.42, -0.011 -0.012, -0.136 14.755, -8.4 29.801, -0.153 0.0561, -0.107 17.353, -7.4532 21.5418	100%	56.25%
29700478	-0.0244 -0.0015 -0.0412 -0.0226 -0.0029 -0.0030 -0.0364 0.0049 -0.0665 0.8976 -0.0042 1.0378 -0.0201 1.2404 0.0200 1.0827 -0.0286 1.2985 -0.0014 0.0079 -0.0016 -0.0044 -0.0033 -0.0012 -0.0001 0.0005 -0.0033 -0.0789 -0.0263 -0.0941 0.0766 -0.0847 -0.0677 -0.1183 0.0127 -0.0966	45%	50%	0.1	-102.823, -102.07 187.306, -0.25 -0.632 0.586 43.279, 128.818 -97.309, -0.24 -0.035, 0.218 38.776, 0.237 -66.122, -0.169	100%	52.50%

					0.305, -0.292 34.109, -47.259 70.536, 0.039 0.051, -0.075 -8.853, -10.271 -5.776		
29700543	0.0000 0.0402 0.0002 0.1102 0.0005 -0.0368 -0.0002 -0.0018 -0.0003 -0.0004 -0.0402 -0.0002 -0.1102 -0.0005 0.0368 0.0002 0.0018 0.0003 -16.5734 -0.0003 -0.0000 -0.0006 -0.0000 -0.0009 -0.0000 -0.0012 -0.0000 -16.5734 -0.0003 -0.0000 -0.0006 -0.0000 -0.0009 -0.0000 -0.0012 -0.0000	95%	90%	0.1	70.3916, 0.0528 -19.5365, 0.2106 -19.6630, 0.1826 -45.3658, 0.0397 1.0793	100%	84.58%
29700707	0.0013 0.0008 0.0024 0.0032 0.0009 -0.0030 0.0015 0.0007 -0.0002 -0.0446 -0.0063 -0.0712 -0.0073 -0.0337 0.0099 -0.0490 -0.0039 -0.0143 -0.0221 0.0029 -0.0222 -0.0008 -0.0306 -0.0074 0.0065 0.0016 -0.0441 0.0125 0.0020 0.0153 0.0005 0.0189 -0.0015 0.0006 -0.0018 0.0155	55%	45%	3	-115.088, -70.063 -199.691, -0.437 -0.061, -0.4817 82.936, 1.642 85.384, 0.417 0.379, 0.312 -82.42, -76.738 -61.477, -0.409 -0.417, -0.552	100%	45%

					79.75, 85.653 106.523, 0.037 -0.093, 0.038 0.11, 24.662 5.338		
29702146	-1.3953 0.0544 0.0000 0.0163 -0.0002 -0.0082 -0.0004 -0.0295 -0.0004 0.0065 0.0016 0.0061 -0.0268 0.0050 0.0169 0.0137 -0.0015 0.0024 -0.0139 -0.0169 -0.0129 0.0215 -0.0113 -0.0099 -0.0256 0.0021 -0.0132 -18.1018 -0.0000 0.0000 -0.0000 0.0000 -0.0000 0.0000 -0.0000 0.0000	80%	61.25%	3	-4.307, -221.018 0.092, 0.321 -12.129, -70.173 0.012, -0.552 -2.859, 97.149 -0.04, -0.678 6.942, 123.513 -0.043, -0.116 7.491, 32.408	100%	60.42%
29702949	-12.3136 -0.0003 -0.0000 -0.0006 -0.0000 -0.0008 -0.0000 -0.0032 -0.0000 0.0070 -0.0014 0.0039 0.0139 0.0128 -0.0135 0.0059 0.0007 0.0037 -0.0044 0.0024 -0.0047 -0.0001 -0.0066 0.0000 -0.0037 -0.0001 -0.0080 0.0200 -0.2155 0.0323 -0.0243 0.0412 0.0032 0.0118 0.0228 0.0705	70%	52.08%	0.5	214.192, 202.525 0.847, 0.287 -175.159, -47.928 -0.006, 0.053 7.43, -3.921 0.106, 0.404 -17.451, -77.876	90%	62.08%

					-0.037, 0.198 2.211, -42.132		
29707204	0.0034 0.0372 0.0097 0.2285 0.0128 -0.2837 -0.0090 0.0390 -0.0634 0.0018 0.0004 0.0013 0.0072 0.0055 0.0018 0.0004 -0.0067 0.0047 -0.0664 -0.0072 -0.0777 -0.0090 -0.1134 0.0080 -0.0422 0.0021 -0.1168 0.5774 -0.2792 -0.0007 -0.0086 -0.0096 -0.0663 0.0030 0.0550 0.0056	65%	54.27%	0.1	-38.984, 80.355 0.153, 0.602 -44.795, -132.923 -0.149, -0.29 38.599, 54.318 -0.093, -0.163 24.883, 41.587 -0.037, 0.183 4.763, -36.278	100%	60.95%
29710601	-0.0840 -0.0034 -0.0480 0.0087 -0.1128 -0.0098 -0.0895 0.0032 -0.0523 -15.9833 -0.0003 -0.0000 -0.0008 -0.0000 -0.0012 -0.0000 -0.0020 -0.0000 0.0046 0.0030 0.0027 -0.0070 0.0062 0.0092 0.0050 -0.0022 0.0029 -15.9833 -0.0003 -0.0000 -0.0008 -0.0000 -0.0012 -0.0000 -0.0020 -0.0000	80%	58.75%	0.1	-215.6719, -186.2963 -0.9927, -0.7731 184.3054, 138.8267 1.1478, 1.0775 -164.9783, -181.7438 -0.4232, -0.5684 33.5772, 108.2577	100%	56.67%

					-0.3001, -0.2262 77.3166, 43.0826		

Table 5.1 User models